VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Anna-Leena Kaunisto

OCR-TEKNOLOGIAA HYÖDYNTÄVÄ ANDROID-APPLIKAATIO GREENSTEP OY:LLE

Tekniikka ja liikenne
2015

ALKUSANAT

Tämä opinnäytetyö on tehty Vaasan ammattikorkeakoulun tietotekniikan koulutusohjelman päättötyönä kevään 2015 aikana Greenstep Oy:lle.

Opinnäytetyön valvojana on toiminut yliopettaja Timo Kankaanpää. Yhteyshenkilönä Greenstepin puolelta toimi markkinointijohtaja Fredrik Teir.

Haluan kiittää Timo Kankaanpäätä ja Fredrik Teiriä sujuvasta yhteistyöstä ja avusta opinnäytetyön tekemisessä.

Vaasassa 12.5.2015

Anna-Leena Kaunisto

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Anna-Leena Kaunisto
Opinnäytetyön nimi	OCR-teknologiaa hyödyntävä Android-applikaatio Greenstep Oy:lle
Vuosi	2015
Kieli	suomi
Sivumäärä	39
Ohjaaja	Timo Kankaanpää

Opinnäytetyö toteutettiin projektina Greenstep Oy:lle. Tarkoituksena oli toteuttaa Android-applikaatio, jonka avulla kuvasta voidaan erotella tekstidataa ja tallentaa se puhelimen muistiin. Sovellus hyödyntää OCR-teknologiaa tekstintunnistukseen.

Työssä toteutettiin Android-sovellus, johon käytettiin Java-ohjelmointikieltä. Lisäksi tekstintunnistusta varten käytettiin C- ja C++ -ohjelmointikielillä toteutettua kirjastoa. Toteutusta varten asetettiin NDK-ympäristö, jonka avulla pystyttiin hyödyntämään kirjastoa sovelluksessa.

Opinnäytetyön tuloksena saatiin toteutettua sovellus, joka tunnistaa tekstiä kuvasta ja tallentaa sen puhelimen muistiin.

Avainsanat Android, Java, Optical Character Recognition

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Anna-Leena Kaunisto
Title	Android application, which uses OCR technology for Greenstep Oy
Year	2015
Language	Finnish
Pages	39
Name of Supervisor	Timo Kankaanpää

Thesis was done as a project for Greenstep Oy. The main purpose of the thesis was to implement an Android application, which extracts text data from images and save the data to phone memory. Application uses OCR technology for text recognition.

Application was implemented using Java as programming language. Text recognition engine uses C and C++ programming languages. When implementing native languages with Java it must be done in Android NDK environment.

The result of this thesis was an application, which extracts text from images and saves the recognized text to phone memory.

Keywords Android, Java, Optical Character Recognition

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

TERMIT JA LYHENTEET

OCR	Optical Character Recognition
XML	Extensible Markup Language
NDK	Native Development Kit
ADT	Android Development Tool
API	Application Programming Interface
JNI	Java Native Interface
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
JSP	JavaServer Pages
DVM	Dalvik Virtual Machine
MVC	Model-View-Controller
SDK	Software Development Kit

1 JOHDANTO

Opinnäytetyön toimeksiantaja on Greenstep Oy. Projektin tarkoituksena on tutkia ja toteuttaa sovellus, joka OCR-teknologian avulla erottelee kuvasta tekstiä ja numeroita. Sovelluksen avulla voidaan helpottaa ja nopeuttaa asiakkaiden kululaskujen käsittelyprosessia. Toteutusvaiheessa tullaan kertomaan sovelluksen ominaisuuksia sekä toteutukseen vaadittavista teknologioista.

Toteutuksen jälkeen applikaation toiminta tullaan testaamaan erilaisten testitapauksien avulla. Lisäksi pohditaan ja perehdytään, mitä rajoituksia sovelluksella on ja kuinka toimintoja tulisi parantaa. Opinnäytetyön lopussa käydään läpi tulevaisuuden suunnitelmat applikaation kehittämisen suhteen ja pohditaan opinnäytetyön onnistuvuutta.

2 GREENSTEP OY

Greenstep Oy on taloushallinnon palveluja tarjoava yritys, joka on perustettu vuonna 2010. Yritys työllistää noin 75 henkilöä Espoossa, Turussa, Tampereella, Oulussa ja Vaasassa.

Greenstepin palveluihin kuuluu muun muassa kirjanpito, palkanmaksu, ostolaskut, myyntilaskut, tilinpäätökset ja veroilmoitukset. Tilitoimistopalvelujen lisäksi yrityksen toimenkuvaan kuuluu esimerkiksi johdon raportointi, corporate finance sekä taloushallinnon prosessikehitys.

Greenstep on yrityksenä poikkeuksellinen, sillä esimerkiksi tilitoimistopalvelu on erikoistunut tukemaan nimenomaan kasvuyrityksiä. Tämä olikin aikanaan yksi tärkeimmistä syistä, miksi Greenstep perustettiin. Yrityksen perustajajäsenten mukaan Suomesta puuttui kasvuyrityksiin erikoistunut palveluntarjoaja, joka pystyy laaja-alaisesti auttamaan yrityksiä heidän talousasioissa. /10/

2.1 Receiptcamera

Receiptcamera on Greenstepin kehittämä applikaatio, joka toimii Android-, iOS- ja Windows-alustoilla. Sovelluksen tarkoituksena on tehdä yritysten kulukorvausanomusten luomisesta, hyväksymisestä ja maksamisesta mahdollisimman helppoa ja nopeaa. Käyttäjä voi ottaa sovelluksella kuvan kuitista sekä lisätä saate- ja tiliöintitiedot, jonka jälkeen tiedot lähetetään suoraan oman yrityksen kirjanpitoon. Sovelluksen pääkäyttötarkoitus on nopeuttaa ja helpottaa sekä asiakkaan että kirjanpitäjän työntekoa. /12/

2.2 Tämänhetkinen tilanne

Receiptcameran myötä Greenstep on huomannut tarpeen applikaatiota tukevalle kuittien kuvaussovellukselle. Receiptcameran kautta lähetettävät kuvat voivat olla tarkkuudeltaan tai valotukseltaan epäselviä, minkä vuoksi kirjanpitäjän on hankala

selvittää menojen suuruutta. Tämän vuoksi Receiptcameran rinnalle halutaan toinen applikaatio, joka mahdollistaa kuvien ottamisen lisäksi rajauksen, muokkauksen ja tekstitietojen kaappaamisen automaattisesti.

2.3 Opinnäytetyön tarkoitus

Tavoitteena on luoda Receiptcameran tueksi toinen applikaatio, joka mahdollistaa kuvien rajauksen, muokkauksen ja tekstitietojen kaappaamisen automaattisesti. Sovelluksen tulisi helppokäyttöinen ja selkeä. Näin ollen käyttäjän ei itse tarvitse kirjoittaa saatetietoja, minkä seurauksena yritys pystyisi tulevaisuudessa yhdistämään sovellusten ominaisuudet. Tuloksena syntyisi aikaasäästävä, kustannustehokas ja kätevä kirjanpitoapplikaatio.

3 KÄYTETYT TEKNOLOGIAT

3.1 Android

Android on Googlen kehittämä, Linux-pohjainen matkapuhelinkäyttöjärjestelmä. Se on suunniteltu pääsääntöisesti käytettäväksi matkapuhelimiin ja tabletteihin. Käyttöjärjestelmän idea perustuu avoimeen lähdekoodiin. Näin ollen laitevalmistajat ja kolmannet osapuolet voivat vapaasti muokata ja levittää sitä. /14/

3.1.1 Arkkitehtuuri

Android-käyttöjärjestelmän tasot sisältävät useita monipuolisia ominaisuuksia. Se sisältää kaikki tarvittavat komponentit, joita käytetään myös yleisessä ohjelmistoympäristössä.

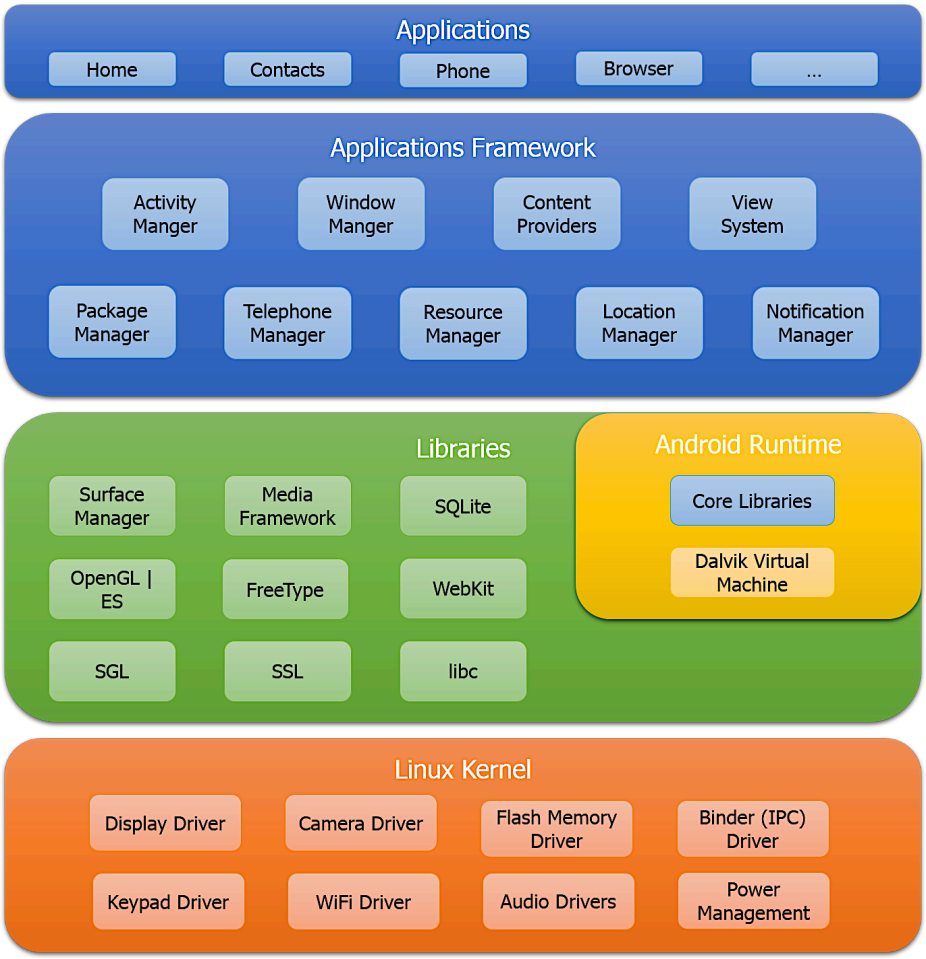
Linux Kernel on koko Android-arkkitehtuurin ydin. Se pitää sisällään erilaisia palveluja, kuten turvallisuus, muistinhallinta, prosessien hallinta ja ajurimallit. Se ei kommunikoi suoraan käyttäjän kanssa, koska se on arkkitehtuuripinossa alimpana.

Kernelin yläpuolella sijaitsee Androidin natiivikirjastot sekä Androidin suoritusvaiheen (Android Runtime). Natiivikirjastoihin lukeutuu esimerkiksi WebKit, SQLite ja FreeType. WebKit vastaa selaintuesta, SQLite hoitaa tietokantayhteyden ja FreeType mahdollistaa erilaisten fonttien käytön.

Suoritusvaihe pitää sisällään ydinkirjastot sekä Dalvik-virtuaalikoneen (Dalvik Virtual Machine, DVM). DVM vastaa Android-sovellusten ajamisesta. DVM on hyvin samanlainen Javan virtuaalikoneen (Java Virtual Machine, JVM) kanssa, mutta se kuluttaa vähemmän muistia ja tarjoaa tehokkaamman suorituskyvyn.

Kirjastojen yllä sijaitsee Android viitekehys. Viitekehyksen avulla voidaan hyödyntää useita laitteiston ominaisuuksia, käyttää paikkatietoja, asettaa muistutuksia ja lisätä hälytyksiä.

Pinon päällimmäisenä sijaitsee sovelluskerros. Kaikki sovellukset ja applikaatiot, kuten kotivalikko ja yhteystiedot sijaitsevat tässä kerroksessa. Nämä hyödyntävät Android viitekehystä, joka puolestaan käyttää Androidin suoritusvaihetta ja kirjastoja. Kaikki edellä mainitut tarvitsevat Kernelin ydinprosesseja toimiakseen muodostaen Android-käyttöjärjestelmän arkkitehtuurin (**Kuvio 1.**). /2/



Kuvio 1. Android-käyttöjärjestelmän arkkitehtuuri. /1/

3.2 Optical Character Recognition

Optical Character Recognition (OCR) on yleisnimi teknologialle, jonka avulla voidaan tunnistaa ja muuntaa erityyppisiä asiakirjoja, kuten PDF- ja kuvatiedostoja, sähköisesti muokattavaan muotoon. Merkintunnistuksessa kootaan skannatusta asiakirjasta havaittuja pikseleitä ja verrataan niiden muodostamia hahmoja olemassa oleviin kirjainhahmoihin ja pyritään näin tunnistamaan oikea merkki, joka voi olla esimerkiksi kirjain, numero tai erikoismerkki. /11/

3.2.1 Tesseract

Tesseract on yksi useista saatavilla olevista OCR-kirjastoista. Sitä voidaan käyttää suoraan sellaisenaan, tai ohjelmistokehityksessä käyttöliittymän avulla. Se on käytössä kaikilla yleisimmillä käyttöjärjestelmillä; Linux, Windows ja Mac OSX. Lisäksi sitä pidetään tällä hetkellä tarkimpana avoimena lähdekoodina olevana kirjastona.

Tesseract tunnistaa kymmeniä eri kieliä, muun muassa englantia, arabiaa, ranskaa ja venäjää. Kielivarastoa on mahdollista laajentaa käyttämällä ulkoisia tiedostoja. /10/

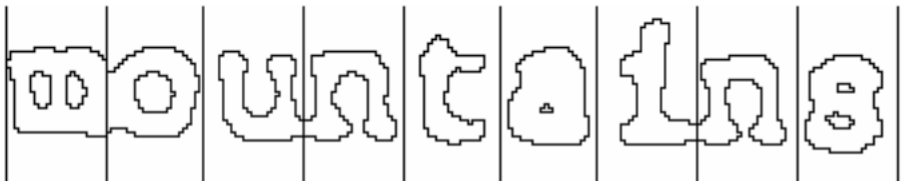
3.2.2 Arkkitehtuuri

Tunnistuskäsittely etenee monivaiheisena prosessina, joista ensimmäinen on yhdistettyjen komponenttien analysointi. Tässä vaiheessa ääriviivat kootaan yhteen ja niistä muodostetaan kimpale.

Kimpaleet järjestetään tekstilinjoille, minkä jälkeen linjat ja alueet analysoidaan kiinteänä alueena tai suhteellisena tekstinä. Tekstilinjat hajotetaan sanoiksi yksitellen riippuen siitä, miten kaukana merkit ovat toisistaan. Kiinteän alueen teksti, missä merkit ovat tasaisesti erillään toisistaan, pilkotaan välittömästi merkkien

solujen toimesta (**Kuvio 2.**). Suhteellisen tekstin välit eivät ole tasaisia, jolloin teksti hajotetaan sanoiksi käyttäen tarkkaa ja epätarkkaa sanaväliä.

Tarkastelemalla ääriviivojen asettelua ja seuraavia ääriviivoja Tesseractin on yksinkertaista havaita myös valkoista tekstiä mustalla pohjalla ja tunnistaa se yhtä helposti kuin mustavalkoinen teksti. Tesseract olikin luultavasti ensimmäinen OCR-kirjasto, joka pystyi käsittelemään käänteistä tekstiä niin yksiselitteisesti.



Kuvio 2. Esimerkki kiinteän alueen sanan pilkkomisesta.

Tämän jälkeen tunnistus etenee kaksivaiheisena välitysprosessina. Ensimmäisessä vaiheessa välitys yritetään tehdä tunnistaen jokainen sana vuorollaan. Jokainen hyväksyttävä sana välitetään fonttien ja merkkien mukaan mukautuvalle luokittelijalle, jonka avulla voidaan tunnistaa epäselviä merkkejä onnistuneesti. Luokittelija saa näin mahdollisuuden tunnistaa tekstin täsmällisemmin.

Koska luokittelija on saattanut huomata tunnistettavan sanan liian myöhään osallistuakseen tunnistukseen alkuvaiheessa, toinen välitys tehdään koko tunnistettavan tekstin läpi. Sanat, joita ei aiemmin voitu tunnistaa tarpeeksi hyvin tunnistetaan uudelleen.

Viimeisessä vaiheessa selvitetään epätarkat sanavälit ja tarkistetaan voidaanko tunnistaa pienikokoista tekstiä. /16/

3.3 Käytetyt ohjelmointikielet

3.3.1 Java

Java on Sun Microsystemsin kehittämä laaja teknologiaperhe ja ohjelmistoalusta, johon kuuluu muun muassa laitteistoriippumaton oliopohjainen ohjelmointikieli sekä ajoaikainen ympäristö virtuaalikoneineen ja luokkakirjastoineen. Java-alusta on käytössä noin 3,8 miljardissa laitteessa matkapuhelimista supertietokoneisiin. /15/

Javaan kuuluu ohjelmointikieli, josta on julkaistu eri versioita kehitysympäristön Java Development Kit (JDK) kehittymisen myötä. Kehitysympäristöön kuuluu kääntäjä ja muut työkalut, kuten jar, javadoc ja jdb. Se sisältää myös täydellisen ajoympäristön Java Runtime Environment (JRE), jota tarvitaan käännettyjen ohjelmien ajamiseen. Ajoympäristö sisältää virtuaalikoneen Java Virtual Machine (JVM), joka on JRE:n versiosta 1.2 lähtien sisältänyt ajonaikaisen käännöksen konekielelle. Ajoympäristöön kuuluvat myös luokkakirjastot, jotka ovat nekin saaneet lisää ominaisuuksia Javan kehittymisen myötä.

Ajoympäristö on nykyisin jaettu kolmeen osaan:

- Java Standard Edition (Java SE) sisältää yleiset ominaisuudet, kuten graafiset käyttöliittymät, tietokanta- ja XML-rajapinnat.
- Java Enterprise Edition (Java EE) on tarkoitettu palvelinsovellusten kehittämiseen ja ajamiseen. Lisäksi se sisältää muun muassa nimeämis- ja hakemistopalvelut, komponenttirajapinnan servlet- ja portlet-määrittelyt, JSP:n (JavaServer Pages) ja muita web-palvelintekniikoita.
- Java Micro Edition (Java ME) on tarkoitettu teholtaan rajoittuneiden laitteiden ohjelmointiin, kuten matkapuhelimiin ja digibokseihin. J2ME eroaa merkittävästi muista ympäristöistä. Se käyttää rajoitetumpaa virtuaalikonetta ja jättää toteuttamatta joitain kielen ominaisuuksia, kuten liukuluvut sekä luokkien purkajat. Java ME on edelleen jaettu konfiguraatioihin ja profiileihin laitteiden ominaisuuksien mukaan. /15/

3.3.2 C

C-kieli on ohjelmointikieli, jota käytetään erilaisten mikroprosessoripohjaisten laitteiden ohjelmointikielenä. C-kielellä voidaan ohjelmoida muun muassa käyttöjärjestelmiä erilaisilla laitteille, kuten tableteille tai puhelimille. /7/

C-ohjelmointikieli on Dennis Ritchien 1970-luvulla luoma, alkujaan UNIX-käyttöjärjestelmän järjestelmäohjelmointia varten luotu ohjelmointikieli. Kieli standardisoitiin myöhemmin, ja onkin nykyään käytetty useiden järjestelmäalustojen kielenä. Monet uudemmat ohjelmointikielet polveutuvat juuri C-kielestä, kuten C++, C# ja Java.

C-kielen etuina pidetään sen siirrettävyyttä ja laiteläheisyyttä. Siirrettävyydellä tarkoitetaan sitä, että lähdekoodia voidaan siirtää sellaisenaan laitteesta toiseen. Tämä johtuu siitä, että C:n piirteet, jotka käyttävät hyväksi kohdelaitteen ominaisuuksia, on eristetty kirjastofunktioihin. Kirjastofunktiot ovat valmiita aliohjelmia, jotka toteutetaan kussakin järjestelmässä laitteiston edellyttämällä tavalla, mutta ovat ohjelmoijan näkökulmasta aina samanlaisia. Laiteläheisyydellä tarkoitetaan sitä, että asioita voidaan toteuttaa tehokkaasti. Laiteläheisyydestä huolimatta C-kieli sisältää korkean tason kielen ohjausrakenteet, joten sillä voidaan kirjoittaa nykystandardien mukaisia ohjelmia. /6/

3.3.3 C++

C++ on ohjelmointikieli, joka sai alkunsa vuonna 1979 nimellä C with Classes eli C-kieli luokilla. Vuonna 1983 sen nimeksi vaihdettiin C++, joka tarkoittaa C-kielessä muuttujan C arvon kasvattamista yhdellä. Sitä käytetään monilla alustoilla ja lukuisissa käyttöjärjestelmissä. C++ on monipuolinen kieli, joka sopii toisaalta matalan tason ohjelmointiin, kuten käyttöjärjestelmien ja laiteajurien kirjoittamiseen ja toisaalta käytännöllisten sovellusten, kuten toimisto-ohjelmien ja pelien tekemiseen.

C++ muistuttaa ulkoisesti hyvin paljon C:tä, ja monet näkevätkin sen vain laajennuksena C-kielestä, johon on lisätty luokat ja merkillinen standardikirjasto. Nämä ja lukuisat muut C++:n lisäykset ovat kuitenkin hyvin olennaisia, ja tekevät itse kielestä monin verroin mutkikkaamman. /8/

3.4 Käytetyt ohjelmistot

Android-sovellusten toteuttamiseen tarvitaan asianmukainen kehitysympäristö. Tässä työssä kehitysohjelmana käytetään Eclipseä, jonka lisätyökaluna on Android-ohjelmoinnin kehitysympäristö ADT. Lisäksi natiivikielien hyödyntämiseen tarvitaan Android NDK-työkalujoukko.

3.4.1 Eclipse

Eclipse on avoimeen lähdekoodiin perustuva ohjelmointiympäristö. Sitä on yleisesti käytetty Java-ohjelmoinnissa, mutta sitä voidaan käyttää myös C/C++ ja PHP –ohjelmointiin. Eclipselle on olemassa useita lisätyökaluja, joiden avulla ohjelmistoympäristöä voi muokata omien tarpeidensa mukaan. /9/

3.4.2 Android Development Tools (ADT)

Android Development Tools (ADT) on Eclipsen liitännäinen, joka laajentaa Eclipsen käyttömahdollisuuksia Android-ohjelmointiin. ADT:n avulla voidaan ohjelmakoodin kirjoittamisen lisäksi luoda käyttöliittymiä, ajaa applikaatioita ja viedä valmiita sovelluksia jakoa varten. /5/

3.4.3 Android Native Development Kit (NDK)

Android NDK on työkalujoukko, joka mahdollistaa sellaisten applikaatioiden toteuttamisen, jotka hyödyntävät C- ja C++ -ohjelmointikieliä. NDK pitää sisällään Javan natiivirajapinnan, Java Native Interfacen (JNI), jonka avulla voidaan kirjoittaa C- ja C++ -ohjelmointikieliä sisältäviä metodeja ja kutsua niitä Java-ohjelmakoodissa. Tekniikassa yhdistyy Android-viitekehyksen helppous yhdessä natiivikielen kirjoituksen kanssa. Toisin sanoen JNI on Javan tapa tukea natiivikoodia. /4/, /13/

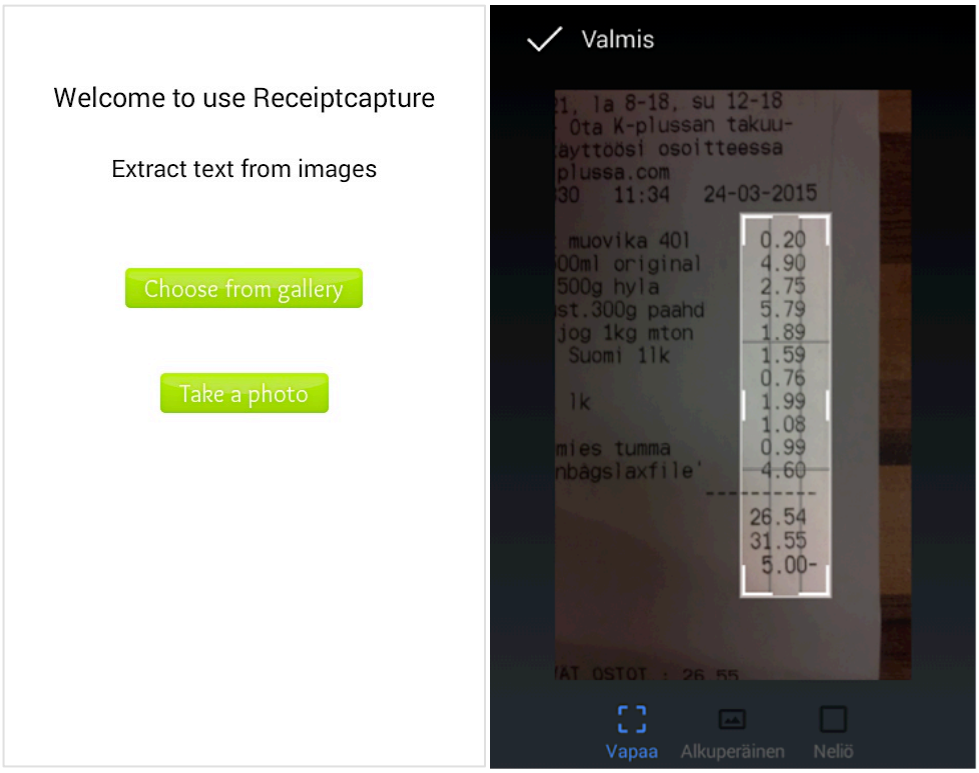
4 SOVELLUKSEN SUUNNITTELU JA MÄÄRITTELY

Sovellus toteutetaan paikallisena applikaationa, ja sillä voidaan eritellä tekstiä valitusta kuva-alueesta. Tämän lisäksi sovelluksella voidaan tallentaa saatu teksti puhelimen muistiin edelleenlähetystä varten.

Sovellus noudattaa mukautettua MVC-mallia (Model-View-Controller), sillä Android-viitekehyksessä sovelluskomponentit eroavat normaalista web-pohjaisesta sovelluksesta monin tavoin. Sovellukset koostuvat neljästä komponenttityypistä: aktiviteeteista (activities), palveluista (services), sisällöntarjoajista (content providers) sekä lähetysten vastaanottajista (broadcast receivers). Komponenttien välinen kommunikointi on pääosin tapahtumapohjaista; eri komponentit eivät keskustele suoraan keskenään, vaan kaikki siirtymät komponenttien välillä tapahtuvat käyttöjärjestelmän välittämien tapahtumaviestien perusteella. Tämän vuoksi Android-sovellukset voivat helposti käyttää toiminnassaan järjestelmän lisäksi toisten sovellusten tarjoamia komponentteja. /3/

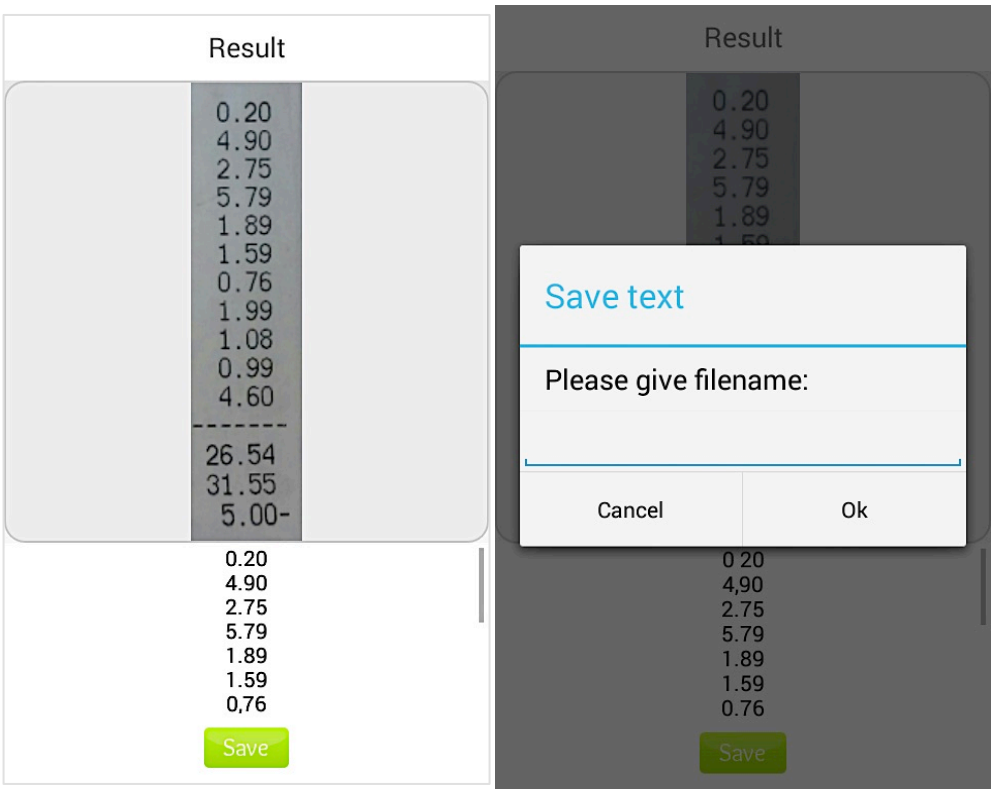
4.1 Toiminnallisuuksien määrittely

Sovelluksen käynnistyessä käyttäjällä on kaksi vaihtoehtoa kuvan valitsemiseen: hän voi joko ottaa kuvan tai valita sen galleriasta. Valinnan jälkeen käyttäjä voi rajata kuvasta haluamansa alueen, josta teksti kaapataan (**Kuvio 3.**). Alue tulisi rajata mahdollisimman tarkasti, jotta vain haluttu teksti voidaan erotella kuvasta. Jos kuvaa ei rajata, kaikki merkit menee OCR:n tunnistuksen läpi, jolloin myös ne merkit, jotka ovat merkityksettömiä, tulevat mukaan tekstitiedostoon.



Kuvio 3. Sovelluksen pääikkuna sekä kuvan rajaus.

Rajattu kuva-alue lähetetään OCR:lle tunnistettavaksi. Tunnistusprosessin päätteeksi tunnistettu teksti näytetään käyttäjälle, minkä jälkeen käyttäjä voi tallentaa tekstin puhelimen muistiin (**Kuvio 4.**).



Kuvio 4. Tulos- ja tallennusnäkymä.

4.2 Vaatimusmäärittely

Vaatimusmäärittelyssä eritellään, mitä sovellukselta vaaditaan. Taulukkoon on koottu toiminnallisuuksia niiden prioriteetin mukaan (**Taulukko 1.**). Taulukosta voidaan selvittää, mitkä toiminnot ovat tärkeitä sovelluksen toimivuuden kannalta.

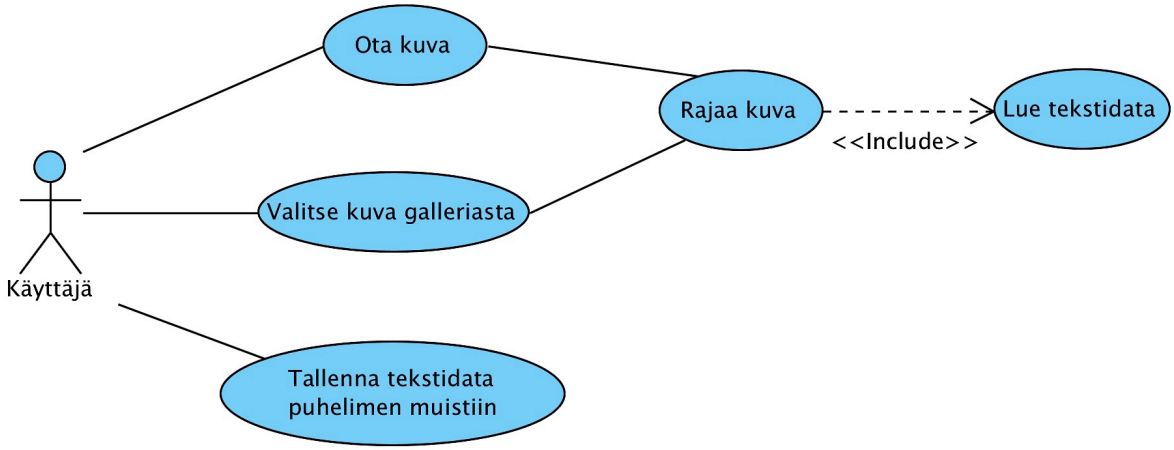
Taulukko 1. Vaatimusmäärittely

Toiminto	**Prioriteetti**
Kuvan ottaminen tai valitseminen galleriasta	1
Tekstitiedon kaappaaminen valitusta kuvasta	1
Kuvan rajaaminen	2
Tekstin tallentaminen tiedostoon	1
Kuvan kirkkauden ja kontrastin säätäminen	3
* Täytyy olla - 1, pitäisi olla - 2, kiva olla - 3.	

4.3 Käyttötapauskaavio

Käyttötapauskaavio kuvaa sovelluksen ja käyttäjän välistä vuorovaikutusta. Siitä näkee sovelluksen sisältämiä tyypillisiä käyttötilanteita.

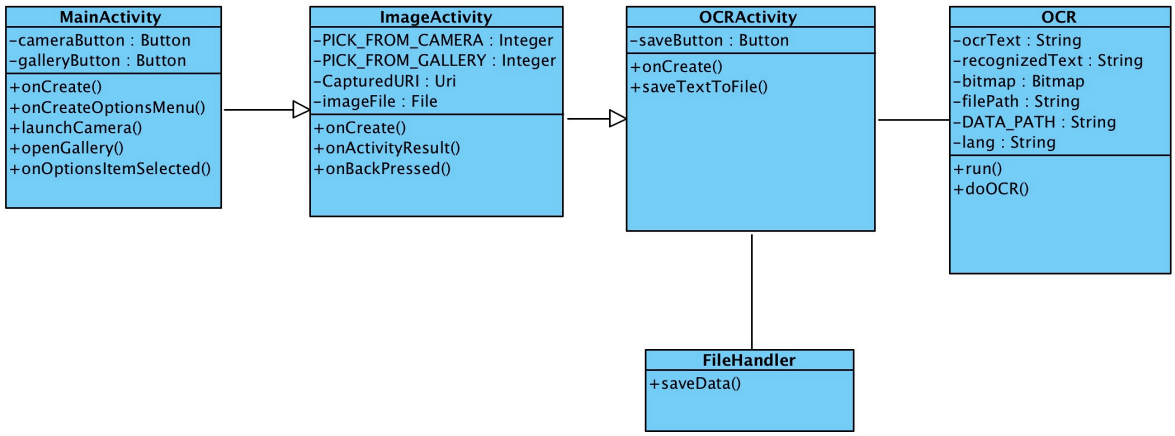
Käyttäjällä on mahdollisuus ottaa kuva kameralla tai valita se puhelimen galleriasta. Tämän jälkeen käyttäjä voi rajata kuvan haluamaansa kokoon, minkä jälkeen tekstidata erotellaan rajatusta alueesta. Lisäksi käyttäjällä on mahdollisuus tallentaa teksti tiedostona puhelimen muistiin (**Kuvio 5.**).



Kuvio 5. Käyttötapauskaavio.

4.3.1 Luokkakaavio

Sovelluksessa tulee olemaan yhteensä 5 luokkaa. MainActivity, ImageActivity ja OCRActivity hoitavat graafisen käyttöliittymän sekä kameran ja gallerian funktiot. OCR-luokka vastaa kuvan kontrastin ja kirkkauden säätämisestä sekä tekstin erottamisesta. FileHandler-luokka on tekstidatan tallennusta varten (**Kuvio 6.**).

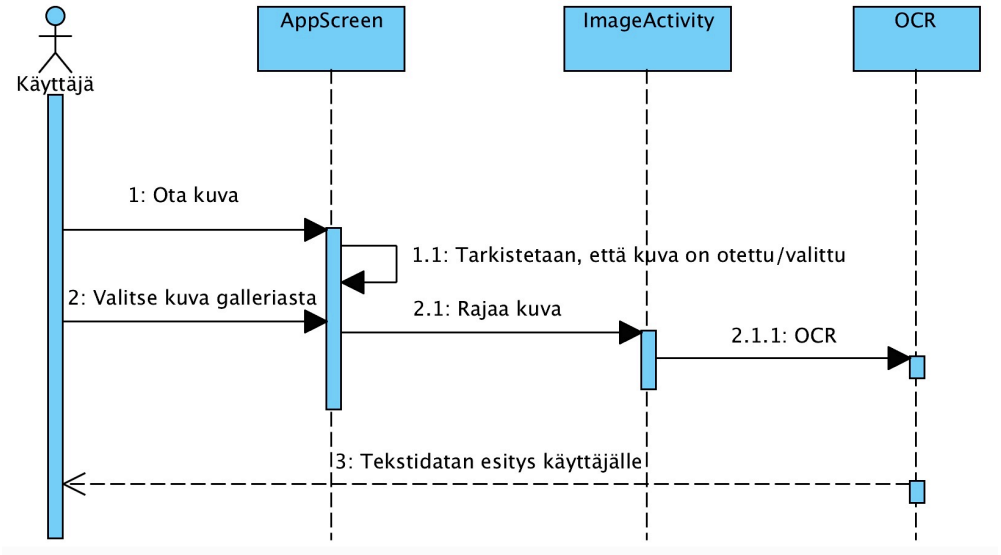


Kuvio 6. Luokkakaavio.

4.3.2 Sekvenssikaavio

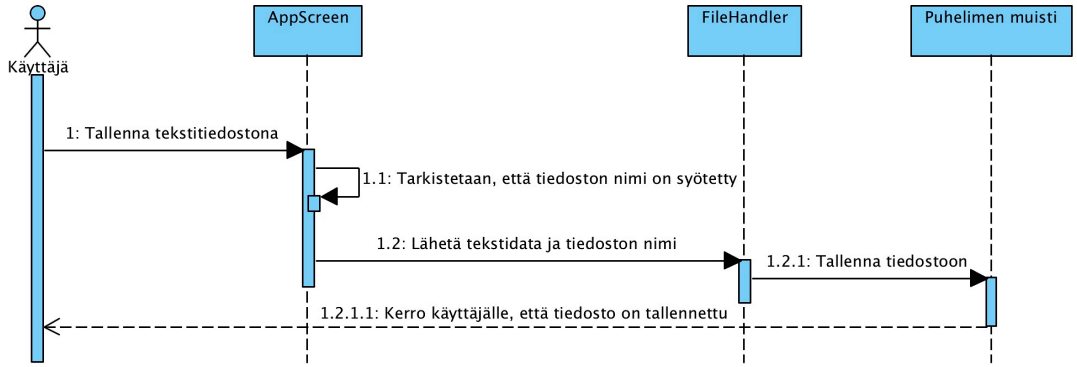
Sekvenssikaavio auttaa havainnollistamaan oliorakenteen toimintaa ja viestien kulkua. Kaaviosta nähdään, kuinka esimerkiksi käyttäjän syöttämät tiedot eli syötteet siirtyvät sovelluksessa.

Käyttäjä valitsee, haluaako ottaa kuvan vai valita sen galleriasta. Tämän jälkeen kuvasta rajataan alue, joka lähetetään OCR:lle. Ennen rajaamista sovellus luonnollisesti tarkistaa, että kuva on valittuna, koska muuten rajaus ei onnistu. OCR:n tunnistuksen jälkeen käyttäjälle esitetään kuvasta eroteltu tekstidata (**Kuvio 7.**).



Kuvio 7. Sekvenssikaavio OCR:n toiminnasta.

Tekstin tallentamista varten käyttäjää pyydetään syöttämään tiedoston nimi. Ennen tallennusta tarkistetaan, että tiedoston nimi on annettu, minkä jälkeen tekstidata lähetetään yhdessä tiedoston nimen kanssa FileHandler-luokalle tallennusta varten. FileHandler-luokka hoitaa tallennuksen puhelimen muistiin, minkä tuloksena käyttäjälle ilmoitetaan, että tallennus on onnistunut (**Kuvio 8.**).



Kuvio 8. Sekvenssikaavio tekstitiedoston tallentamisesta.

5 SOVELLUKSEN TOTEUTUS

5.1 Yleistä toteutuksesta

Työssä toteutetaan sovellus, joka erottelee kuvasta tunnistamansa merkit. Merkit voivat olla tekstiä, numeroita tai muita merkkejä, esimerkiksi välimerkki. Sovellukseen toteutetaan ominaisuudet, joilla kuva otetaan kameralla tai valitaan galleriasta, rajataan ja erotellaan merkit. Lisäominaisuuksina kuvan kirkkautta ja kontrastia manipuloidaan, jotta tunnistustulos olisi paras mahdollinen. Tämän lisäksi eroteltu teksti voidaan tallentaa tiedostona puhelimen muistiin lähetystä varten.

Toteutus eteni kokeilemalla eri tapoja, sillä ei ollut aina varmuutta, mitä tekniikoita sovellus vaatii ja pystytäänkö se toteuttamaan. Aiheesta löytyi paljon informaatiota ja ohjeistusta, mutta usein ne olivat ristiriidassa toistensa kanssa. Tämän vuoksi aika ajoin oli haastavaa löytää informaatiota, joka toimisi omaan tarkoitukseen parhaiten.

5.2 Android NDK

Natiivikehitysympäristön (Native Development Kit, NDK) -ympäristön asetus on kokonaisuudessaan hyvin laaja työvaihe. Ohjeistusta ympäristön pystyttämiseen löytyy paljon, mutta ne ovat myös ristiriidassa toistensa kanssa. Tämän vuoksi oikean ohjeistuksen löytäminen oli hankalaa, ja työvaihe kesti pitkään. Asennusprosessi on myös erilainen eri käyttöjärjestelmille. Tässä työssä ympäristönä toimii Mac OS X Yosemite.

Ennen varsinaista asennusta on oltava käytössä Eclipse ADT (Android Development Tool). ADT:n yhteydessä tulee muun muassa Android SDK-työkalu (Software Development Kit), Android Platform –työkalu sekä viimeisin Androidin sovellusalusta. Kun ADT on käytössä, voidaan asentaa NDK.

NDK-työkalu tulee pakattuna kansiona. Avatakseen tiedoston on komentorivillä navigoitava tiedostopolkuun ja annettava komento ”chmod a+x <tiedoston ni-

mi>”. Chmod tulee sanoista change mode, ja sitä käytetään tiedostojen käyttäjäoikeuksien muuttamiseen. Parametri a tulee sanasta all ja x sanasta execute. Toisin sanoen komennolla lisätään kaikille käyttäjille suoritustila tiedostoon. Kun tiedostolle on asetettu suoritustila, voidaan antaa komento ”./<tiedoston nimi>”, jolloin kansio purkaa itsensä.

Seuraavana on asetettava tarvittavat ympäristömuuttujat. Tämä vaihe on kriittinen, sillä asennuksen seuraavat vaiheet eivät onnistu ilman ympäristömuuttujien oikeanlaista määritystä. OS X –käyttöjärjestelmässä ympäristömuuttujat määritellään .bash_profile –tiedostoon. Tiedoston muokkaus tapahtuu komennolla ”nano .bash_profile”. Ympäristömuuttujat on asetettava sekä Android SDK:lle että Android NDK:lle (**Kuvio 9.**).

```
ANDROID_SDK="SDK:n tiedostopolku"
ANDROID_NDK="NDK:n tiedostopolku"

PATH="$PATH:$ANDROID_SDK/tools:$ANDROID_SDK/platform-tools:$ANDROID_NDK"

export ANDROID_SDK
export ANDROID_NDK
export PATH
```

Kuvio 9. NDK-ympäristömuuttujien määrittely.

Tämän jälkeen tiedosto tallennetaan ja siirrytään konfiguroimaan Eclipseä. Ensiksi Tesseract on määriteltävä projektin kirjastoksi, minkä jälkeen projektiin luodaan jni-niminen kansio. Kansioon generoidaan myöhemmin projektin natiivitiedostot. Lisäksi kansioon on luotava Android.mk –niminen tiedosto, jolla kerrotaan mitä natiiviresursseja projekti käyttää. Toisin sanoen se on Android-ympäristössä käytettävä makefile-tiedosto.

Viimeisenä työvaiheena on jni-kansion sisältämien natiivitiedostojen kääntäminen. Tämä tapahtuu komentorivillä, projektin tiedostopolussa. Kääntäminen ta-

pahtuu komennolla “ndk-build”, jonka ansiosta natiivitiedostot generoituvat Android-ympäristön käytettäväksi.

5.3 Kuvan valitseminen

Kuvan valitsemiseen on kaksi tapaa: se voidaan ottaa kameralla tai valita galleriasta. Valinta tapahtuu MainActivity-luokassa määriteltyjen nappien kautta. Activity-luokat ovat nimensä mukaisesti aktiviteetteja kuvaavia luokkia. Ne kommunikoivat käyttäjän ja sovelluksen muiden luokkien kanssa.

Kummallekin napille on luotu metodi, joka suorittaa käyttäjän valitseman toiminnon. Molemmissa metodeissa luodaan Intent-olio sen mukaan, minkä valinnan käyttäjä on tehnyt. Intent-olioita käytetään aina, kun halutaan välittää mitä tahansa informaatiota aktiviteetilta toiselle. Lopuksi siirrytään ImageActivity-luokkaan suorittamaan tätä toimintaa (**Kuvio 10.**).

```
// Method to launch camera
public void launchCamera() {

    // Set camera action to ImageActivity
    Intent cameraIntent = new Intent(MainActivity.this, ImageActivity.class);
    cameraIntent.putExtra("selection", 1);

    //Start ImageActivity
    startActivity(cameraIntent);
    overridePendingTransition(android.R.anim.slide_in_left,
            android.R.anim.slide_out_right);

}

// Method to open gallery
public void openGallery() {

    // Set gallery action to ImageActivity
    Intent galleryIntent = new Intent(MainActivity.this, ImageActivity.class);
    galleryIntent.putExtra("selection", 2);

    //Start ImageActivity
    startActivity(galleryIntent);
    overridePendingTransition(android.R.anim.slide_in_left,
            android.R.anim.slide_out_right);
}
```

Kuvio 10. Intent-olioiden luominen ja välitys ImageActivity-luokalle.

Itse kuvan valitseminen tapahtuu ImageActivity-luokassa. Luokalle välitetään valinta sen mukaan, kumpaa MainActivity-luokan nappia on painettu. Sekä kameran että gallerian käyttö on toteutettu Androidin standarditoiminnoilla, jolloin Intent-olio avaa kameran tai gallerian syöttöarvoilla ”ACTION_IMAGE_CAPTURE” tai ”ACTION_PICK” (**Kuvio 11.**).

```
// if camera action is chosen
if (choice == 1) {
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    cameraIntent.putExtra("crop", "true");
    String fileName = "temp.jpg";
    ContentValues values = new ContentValues();
    values.put(MediaStore.Images.Media.TITLE, fileName);
    CapturedURI = getContentResolver().insert(
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values);

    // try to start cameraIntent
    try {
        cameraIntent.putExtra("return-data", true);
        cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, CapturedURI);
        startActivityForResult(cameraIntent, PICK_FROM_CAMERA);

    } catch (ActivityNotFoundException e) {
        e.printStackTrace();
    }
}

// If gallery action is chosen
if (choice == 2) {
    Intent galleryIntent = new Intent(Intent.ACTION_PICK,
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    galleryIntent.putExtra("crop", "true");
    galleryIntent.putExtra("scaleUpIfNeeded", "true");

    // try to start galleryIntent
    try {
        galleryIntent.putExtra("return-data", true);
        startActivityForResult(galleryIntent, PICK_FROM_GALLERY);

    } catch (ActivityNotFoundException e) {
        e.printStackTrace();
    }
}
```

Kuvio 11. Kamera- ja galleriatoimintojen määrittely.

Intent-olion palauttama kuvadata käsitellään sen mukaan, miten kuva on valittu. Tässä tapauksessa lähetetään kuva, joka saadaan olion Intent.putExtra-ominaisuudesta. Kuvadatasta muodostetaan olio, jolla kuva esitetään käyttäjälle. Kuvasta tarvitaan myös sen tiedostopolku, joka lähetetään myöskin olion mukana (**Kuvio 12.**).

```
// create bitmap object from image
if (extras != null) {

    final Bitmap galleryBitmap = extras.getParcelable("data");
    Intent OCRIntent = new Intent(ImageActivity.this,
            OCRActivity.class);
    OCRIntent.putExtra("BitmapImage", galleryBitmap);
    OCRIntent.putExtra("filePath", picturePath);

    // Send bitmap to ocr
    setResult(RESULT_OK, OCRIntent);
    startActivity(OCRIntent);
    overridePendingTransition(android.R.anim.slide_in_left,
            android.R.anim.slide_out_right);
    ImageActivity.this.finish();
}
```

Kuvio 12. Bitmap-olion muodostaminen kuvadatasta.

5.4 Kuvan manipulointi

OCR:n tulosta voidaan parantaa merkittävästi muokkaamalla kuvan kontrastia ja kirkkautta ennen tunnistusprosessia. Muokatussa kuvassa olevat merkit erottuvat taustasta selkeämmin, jolloin tunnistus on helpompaa ja nopeampaa. Manipulointi tapahtuu taustalla, jolloin kontrastin muutos ei näy käyttäjälle.

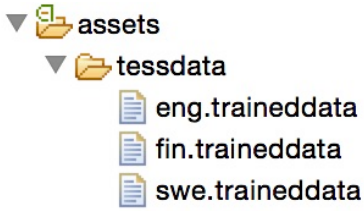
Rajatusta Bitmap-oliosta otetaan kopio, joka lähetetään myöhemmin tunnistusta varten. Muokkaus tapahtuu matriisilla, johon määritellään kirkkaudelle ja kontrastille yksilölliset arvot. Matriisi piirretään Canvas-olion avulla kuvaan ikään kuin suodattimena kuvan päälle (**Kuvio 16.**). Käyttäjä ei näe tätä toimenpidettä, vaan hänelle näytetään alkuperäinen kuva.

```
// Adjust brightness and contrast
Bitmap newOCRbitmap = bm.copy(Bitmap.Config.ARGB_8888 ,true);
Canvas canvas = new Canvas(newOCRbitmap);
Paint paint = new Paint();
ColorMatrix cm = new ColorMatrix();

// Set brightness and contrast values
float contrast = 3.0f;
float brightness = -173f;

// Set ColorMatrix
cm.set(new float[] {
        contrast, 0, 0, 0, brightness,
        0, contrast, 0, 0, brightness,
        0, 0, contrast, 0, brightness,
        0, 0, 0, 1, 0});

// Paint matrix on image
paint.setColorFilter(new ColorMatrixColorFilter(cm));
Matrix matrix = new Matrix();
canvas.drawBitmap(newOCRbitmap, matrix, paint);
```

Kuvio 13. Kontrastin ja kirkkauden muokkaaminen.

5.5 OCR

Kun kuva on valittu ja rajattu, se välitetään OCR-luokalle oliona. Luokassa määritellään muun muassa kielet, joita ohjelmalla halutaan tunnistaa. Kieliä varten assets-kansioon sisällytetään ulkoiset kielitiedostot. Ennen tekstintunnistusprosessin aloittamista nämä tiedostot täytyy kopioida käytettävään laitteeseen, jos halutaan tunnistaa englannin kielen lisäksi muita kieliä (**Kuvio 12.**).

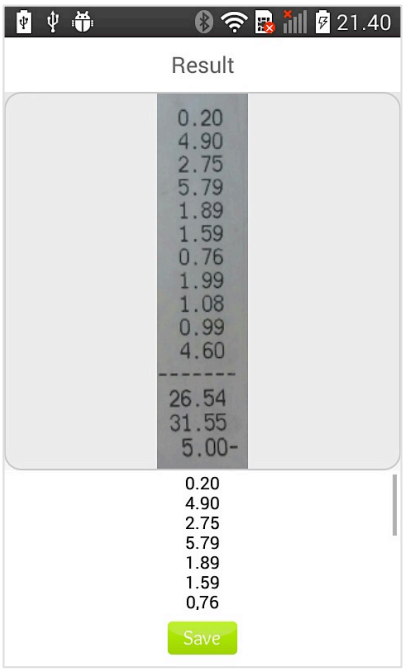


Kuvio 14. OCR:n käytössä olevat kielitiedostot.

Varsinainen tekstintunnistus suoritetaan TessBaseAPI-oliolla, joka on johdettu käytettävästä Tesseract-kirjaston Java-luokasta. Oliolle alustetaan aiemmin kopioidut kielitiedostot ja määritellään lisäksi sallitut ja ei-sallitut merkit. Näin voidaan rajata tiettyjä merkkejä tekstitunnistuksen ulkopuolelle tai päinvastoin sallia tiettyjen merkkien tunnistamisen. Lisäksi tunnistettavan tekstin muuttuja alustetaan OCR-kirjaston vaatimaan UTF8-tekstiformaattiin. Lopuksi tunnistettu teksti liitetään recognizedText-muuttujaan, joka palautetaan OCRActivity-luokalle ja esitetään yhdessä kuvan kanssa käyttäjälle (**Kuviot 13. ja 14.**).

```
//OCR engine
TessBaseAPI baseApi = new TessBaseAPI();
baseApi.setDebug(true);
baseApi.init(DATA_PATH, "eng");
baseApi.init(DATA_PATH, "fin");
baseApi.init(DATA_PATH, "swe");
baseApi.setImage(newOCRbitmap);
String recognizedText = baseApi.getUTF8Text();
baseApi.end();

//Set approved characters and non approved characters to English
if (lang.equalsIgnoreCase("eng")) {
    String whiteList = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz1234567890,.%-";
    String blackList = "üûÜÜíÌìÍßøáÀàÁïÏéèÈØø<>#?ääö;:/()";
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, whiteList);
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, blackList);

    recognizedText = recognizedText.replaceAll(blackList, ""); // remove space
}

//Set approved characters and non approved characters to Finnish and Swedish
if (lang.equalsIgnoreCase("fin") || lang.equalsIgnoreCase("swe")) {
    String whiteList = "ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖabcdefghijklmnopqrstuvxyzääö1234567890,.%-";
    String blackList = "üûÜÜíÌìÍßøáÀàÀïÏéèÈØø<>#?;:/()";
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, whiteList);
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, blackList);

    recognizedText = recognizedText.replaceAll(blackList, "");// remove space
}

// Return recognized text to OCRActivity
recognizedText = recognizedText.trim();
return recognizedText;
```

Kuvio 15. Tekstintunnistus sekä sallittujen merkkien määrittely.



Kuvio 16. Tuloksen esittäminen käyttäjälle.

5.6 Tekstin tallennus

Käyttäjällä on mahdollisuus tallentaa kaapattu teksti puhelimen muistiin tekstitiedostona. Save-nappia painettaessa käyttäjältä pyydetään tiedoston nimi, minkä seurauksena tiedosto tallennetaan puhelimen sisäiseen muistiin FileHandler-luokan saveData-metodin avulla. Metodin palautusarvolla ”true” tai ”false” kerrotaan käyttäjälle, onko tallennus onnistunut (**Kuvio 15.**).

```
// write text data to internal memory
public boolean saveData(String recognizedText, EditText input) throws IOException {

    String TAG = "FileHandler";
    String filepath = "/storage/emulated/0/receiptcapture/Receiptfiles/";
    File myFile = new File(filepath + input.getText() + ".txt");
    File dir = new File(filepath);

    //create directory if necessary
    if (dir.mkdir()) {
        Log.i(TAG, "Directory created");
    } else {
        Log.i(TAG, "Directory is not created");
    }

    //write file
    if (!myFile.exists()) {
        myFile.createNewFile();
        FileOutputStream fOut = new FileOutputStream(myFile);
        OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
        myOutWriter.append(recognizedText);
        myOutWriter.close();
        fOut.close();
        return true;
    } else {
        return false;
    }
}
```

Kuvio 17. Tekstin tallennus puhelimen muistiin.

6 SOVELLUKSEN TESTAUS

6.1 Toiminnot

Testaus tehtiin aina, kun jokin osa sovelluksesta saatiin toteutettua. Kun sovelluksen oleelliset toiminnot olivat valmiina, pystyttiin testaamaan myös OCR:n toimintaa. Toteutuksen jälkeen suoritettiin testitapauksia, joiden avulla pystytään takaamaan tärkeimpien toimintojen onnistuvuus sekä mahdolliset virheet (**Taulukko 2**.).

Taulukko 2. Testitapaukset

ID	Testitapaukset	Odotettu tulos	Tulos
A1	Kuvan ottaminen kameralla tai valitseminen galleriasta sekä sen rajaaminen	Kuva voidaan ottaa kameralla tai valita galleriasta, ja se voidaan rajata	OK
A2	Tekstintunnistus rajatusta kuva-alueesta.	OCR käsittelee valitun kuvan ja tunnistaa siinä olevat merkit	OK
A3	Tekstin tallentaminen puhelimen muistiin	Tiedosto tallennetaan puhelimeen	OK

6.2 Rajoitukset

Käyttäjä voi itse osittain vaikuttaa tekstintunnistuksen tarkkuuteen. Tuloksen varmuuteen vaikuttavat muun muassa miten kuva on otettu ja millaisessa kunnossa kuitti on. Jos kuitti on hyvässä kunnossa eikä se ole päässyt haalistumaan, tunnistustulos on todennäköisemmin tarkempi. Edellä mainittujen asioiden lisäksi kuvan tulisi olla mahdollisimman tarkasti ja läheltä otettu sekä tunnistettavan

tekstin suorassa, jotta jokainen merkki olisi mahdollisimman selkeästi erotettavissa.

Sovelluksessa on myös rajoituksia, joihin käyttäjä ei voi suoranaisesti vaikuttaa. Esimerkiksi kuitin fonttikoko, asettelu ja musteen tummuus ovat ominaisuuksia, joihin on mahdotonta vaikuttaa, mutta ovat kriittisiä tunnistuksen kannalta. Lisäksi kuvan tarkkuus ja koko ovat seikkoja, joita ei suoranaisesti voi muuttaa. Koska OCR:n toiminta perustuu pikseleiden tunnistamiseen, on luonnollista, että pikseleiden määrä vaikuttaa myös tunnistustulokseen.

7 YHTEENVETO

Opinnäytetyön tuloksena saatiin toimiva prototyyppi, joka käyttää hyväkseen OCR-teknologiaa. Kaikki tärkeimmät ominaisuudet saatiin toteutettua, joita on kuvan ottaminen ja valitseminen, kuvan rajaaminen ja tekstin tunnistaminen.

Työ oli hyvin haastava, sillä en ollut aiemmin kuullut kyseisestä teknologiasta. Tämän vuoksi oli opeteltava paljon uutta, jotta sovelluksen toteutus olisi mahdollista. Lisäksi omia ohjelmointitaitoja oli laajennettava, sillä Android-ohjelmointi tai natiivikoodin hyödyntäminen Java-ohjelmistotuotannossa ei ollut minulle tuttua. Tämän vuoksi alussa aikaa tuntui menevän paljon pelkästään tutkimus- ja pohjatyön tekemiseen. Loppujen lopuksi voidaan todeta, että työ oli todella mielenkiintoinen ja opetti muun muassa pitkäjänteisyyttä, itsenäistä työntekoa sekä pienempien osakokonaisuuksien hallitsemista ja niiden vaikutusta koko sovellukseen.

Tulevaisuutta ajatellen, sovellusta tulisi kehittää paremmaksi. Toteutuksen aikana testattiin ympäristöä aina kun saatiin jokin toiminto valmiiksi. Tämän myötä tuli huomattua, että NDK-ympäristöä tulisi vakauttaa ja tunnistustulokseen vaikuttavia tekijöitä pitäisi tunnistaa, jonka seurauksena sovellusta voitaisiin kehittää eteenpäin. OCR:n toiminta tuntui aika ajoin riippuvan päivästä: joinakin päivinä tekstintunnistuksen tulokset eivät olleet kunnollisia, kun taas seuraavalla testauskerralla tunnistusprosessi toimi erinomaisesti.

LÄHTEET

/1/ Android arkkitehtuuri, Viitattu 7.4.2015

https://lokeshv.wordpress.com/2014/06/26/android-architecture/

/2/ Android arkkitehtuurin osat, Viitattu 7.4.2015

http://www.javatpoint.com/android-software-stack

/3/ Android-sovelluksen arkkitehtuuri, Viitattu 14.4.2015

https://helda.helsinki.fi/bitstream/handle/10138/43189/gradu.pdf?sequence=2

/4/ Android NDK, Viitattu 21.3.2015

https://developer.android.com/tools/sdk/ndk/index.html

/5/ Android Development Tools (ADT), Viitattu 13.4.2015

http://developer.android.com/tools/sdk/eclipse-adt.html

/6/ C-ohjelmointikielen hyödyt, Viitattu 7.4.2015

http://cs.stadia.fi/~silas/ohjelmointi/c_opas-C-kielen.html

/7/ C-ohjelmoinnin perusteet, Viitattu 7.4.2015

http://ideaport.edu.hel.fi/C/Ckieli/sivuc1.html

/8/ C++ opas, Viitattu 7.4.2015

http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=cpp_ohj_01#cjac

/9/ Eclipse ohjelmointiympäristö, Viitattu 13.4.2015

https://www.eclipse.org/ide/

/10/ Greenstep Oy, Viitattu 20.3.2015

www.greenstep.fi

/11/ OCR - Optical Character Recognition, Viitattu 20.3.2015

https://help.ubuntu.com/community/OCR

/12/ Receiptcamera, yleiskuvaus, Viitattu 13.4.2015

http://receiptcamera.fi/meista/

/13/ What is Android Native Development, Viitattu 21.3.2015

http://www.electronicsweekly.com/eyes-on-android/what-is/what-is-android-native-code-2011-11/

/14/ What is Android, Viitattu 4.5.2015

http://heavy.com/tech/2013/06/what-is-android-os-operating-system-info-wiki/

/15/ Yleistietoa Javasta, Viitattu 3.4.2015

http://www.webopas.net/mika_java.html

/16/ Yleiskatsaus Tesseract OCR –kirjastoon, Viitattu 23.3.2015

http://www.helsinki.fi/~mpsilfve/ocr_course/materials/tesseracticdar2007.pdf